Efficient Algorithms for Optimal Perimeter Guarding

Si Wei Feng*, Shuai D. Han*, Kai Gao† and Jingjin Yu*
*Department of Computer Science, Rutgers University at New Brunswick, Piscataway, New Jersey, U.S.A.
† School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui, China

***Abstract*—We investigate the problem of optimally assigning a large number of robots (or other types of autonomous agents) to guard the perimeters of closed 2D regions, where the perimeter of each region to be guarded may contain multiple disjoint polygonal chains. Each robot is responsible for guarding a subset of a perimeter and any point on a perimeter must be guarded by some robot. In allocating the robots, the main objective is to minimize the maximum 1D distance to be covered by any robot along the boundary of the regions. For this optimization problem which we call optimal perimeter guarding (OPG), thorough structural analysis is performed, which is then exploited to develop fast exact algorithms that run in guaranteed low polynomial time. In addition to formal analysis and proofs, experimental evaluations and simulations are performed that further validate the correctness and effectiveness of our algorithmic results.**

I. Introduction

Consider the scenario from Fig. 1, which contains a closed region with its boundary or border demarcated by the red and dotted blue polygonal chains (p-chains for short). To secure the region, either from intrusions from the outside or unwanted escapes from within, it is desirable to deploy a number of autonomous robots to monitor or guard either the entire boundary or selected portions of it, e.g., the three red p-chains), with each robot responsible for a continuous section. Naturally, one might also want to have an even coverage by the robots, e.g., minimizing the maximum effort from any robot. In practice, such effort may correspond to sensing ranges or motion capabilities of robots, which are always limited. As an intuitive example, the figure may represent the top view of a castle with its entire boundary being a high wall on which robots may travel. The portion of the wall marked with the three red p-chains must be protected whereas the part marked by the dotted blue p-chains may not need active monitoring (e.g., the outside of which may be a cliff or a body of deep water). The green and orange p-chains show an optimal distribution of the workload by 8 robots that covers all red p-chains but skips two of the three blue dotted p-chains.

More formally we study the problem of deploying a large number of robots to guard a set of 1D *perimeters*. Each perimeter is comprised of one or more 1D (p-chain) *segments* that are part of a circular boundary (e.g., the red p-chains in Fig. 1). Each robot is tasked to guard a continuous 1D p-chain that *covers* a portion of a perimeter. As the main objective, we seek an allocation of robots such that *(i)* the union of the robots' coverage encloses all perimeters and *(ii)* the maximum coverage of any robot is minimized. We call this 1D deployment problem the Optimal Perimeter Guarding (OPG) problem.

In this work, three main OPG variants are examined. The settings regarding the perimeter in these three variants are: *(i)* multiple perimeters with each having a single connected component; *(ii)* a single perimeter containing multiple connected components; and *(iii)* multiple perimeters with each containing multiple connected components (the most general case). For all three variants, we have developed exact algorithms for solving OPG that runs in low polynomial time. More specifically, let there be n robots, m perimeters, with perimeter i $(1 \le i \le m)$ containing q_i connected components. If $m = 1$, then let the only perimeter contains q connected components. For the three variants, our algorithm computes an optimal solution in time $O(m(\log n + \log m) + n)$, $O(q^2 \log(n + q) + n)$, and $O((\sum_{1\le i\le m} q_i^2)\log(n+\sum_{1\le i\le m} q_i)+n)$, respectively, which are roughly quadratic in the worst case. The modeling of the OPG problem and the development of the efficient algorithms for OPG constitute the main contribution of this paper.

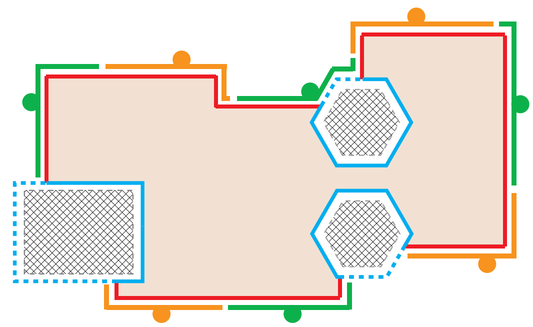

Fig. 1: An illustrative scenario where a perimeter, in this case represented as the red polygonal chains (p-chains), must be guarded by $n = 8$ robots, which are constrained to only travel along the perimeter boundary (the red p-chains plus the dotted blue ones, which are gaps that do not need to be guarded). An optimal set of locations for the 8 robots and the coverage region for each robot are marked on the perimeter boundary in green and orange, which minimizes the maximum coverage required for any robot.

With an emphasis on the deployment of a large number of robots, within multi-robot systems research [1]–[4], our study is closely related to *formation control*, e.g., [5]–[11], where the goal is to achieve certain *distributions* through continuous (often, local sensing based) interactions among the agents or robots. Depending on the particular setting, the distribution in question may be spatial, e.g., rendezvous [5], [11], or maybe an agreement in agent velocity is sought [6], [8]. In these studies, the resulting formation often has some degree-of-freedoms left unspecified. For example, rendezvous results [5], [11] often come with exponential convergence guarantee, but the location of rendezvous is generally unknown *a priori*.

On the other hand, in multi-robot task and motion planning problems (e.g., [12]–[19]), especially ones with a *task allocation* element [12], [14]–[17], [19], the (permutation-invariant) target configuration is often mostly known. The goal here is finding a one-to-one mapping between individual robots and the target locations (e.g., deciding a *matching*) and then plan (possibly collision-free) trajectories for the robots to reach their respective assigned targets [16], [17], [19]. In contrast to formation control and multi-robot motion planning research, our study of OPG seeks to determine an exact, optimal distribution pattern of robots (in this case, over a fairly arbitrary, bounded 1D topological domain). Thus, solutions to OPG may serve as the target distributions for multi-robot task and motion planning, which is the main motivation behind our work. The generated distribution pattern is also potentially useful in multi-robot persistent monitoring [20] and coverage [21], [22] applications, where robots are asked to carry out sensing tasks in some optimal manner.

As a multi-robot coverage problem, OPG is intimately connected to Art Gallery problems [23], [24], with origins traceable to half a century ago [25]. Art Gallery problems assume a visibility-based [26] sensing model; in a typical setup [23], the *interior* of a polygon must be visible to at least one of the guards, which may be placed on the boundaries, corners, or the interior of the polygon. Finding the optimal number of guards are often NP-hard [27]. Alternatively, disc-based sensing model may be used, which leads to the classical *packing* problem [28], [29], where no overlap is allowed between the sensors' coverage area, the *coverage* problem [30]–[34], where all workspace must be covered with overlaps allowed, or the *tiling* problem [35], where the goal is to have the union of sensing ranges span the entire workspace without overlap. For a more complete account on Art Gallery, packing, and covering, see Chapters 2, 3, and 33 of [36]. Despite the existence of a large body of literature performing extensive studies on these intriguing computational geometry problems, these types of research mostly address domains that are 2D and higher. To our knowledge, OPG, as an optimal coverage problem over a non-trivial 1D topological space, represents a practical and novel formulation yet to be fully investigated.

The rest of the paper is organized as follows. The OPG problem and some of its most basic properties are described in Section II. In Section III, a thorough structural analysis of OPG with single and multiple perimeters is performed, paving the way for introducing the full algorithmic solutions in Section IV. Then, in Section V, comprehensive numerical evaluations of the multiple polynomial-time algorithms are carried out. In addition, two realistic application scenarios are demonstrated. In Section VI, we conclude with additional discussions.

II. The Optimal Perimeter Guarding Problem

Let $\mathcal{W} \subset \mathbb{R}^2$ be a compact (i.e., closed and bounded) two-dimensional workspace. There are m pairwise disjoint *regions* $\mathcal{R} = \{R_1, \ldots, R_m\}$ where each region $R_i \subset \mathcal{W}$ is homeomorphic to the closed unit disc, i.e., there exists a continuous bijection $f_i : R_i \to \{(x, y) \mid x^2 + y^2 \leq 1\}$ for all $1 \leq i \leq m$. For a given region R_i, let ∂R_i be its (closed) boundary (therefore, f_i maps ∂R_i to the unit circle $\mathbb{S}^1$). With a slight abuse of notation, define $\partial\mathcal{R} = \{\partial R_1, \ldots, \partial R_m\}$. For each $\mathcal{R}_i$, $P_i \subset \partial R_i$ is called the *perimeter* of R_i which is either a single closed curve or formed by a finite number of possibly curved line segments. In this paper, we assume a perimeter is given as a single p-chain (possibly a polygon) or multiple disjoint p-chains. Let $\mathcal{P} = \{P_1, \ldots, P_m\}$, which must be *guarded*. More formally, each P_i is homeomorphic to a compact subset of the unit circle. For a given P_i, each of its maximal connected component (a p-chain) is called a *perimeter segment* or *segment*, whereas each maximal connected component of $\partial R_i \backslash P_i$ is called a *perimeter gap* or *gap*. An example is illustrated in Fig. 2 with two regions.

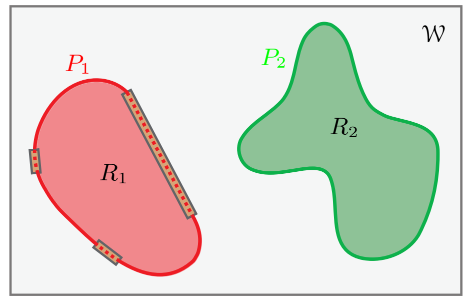

Fig. 2: An example of a workspace $\mathcal{W}$ with two regions $\{R_1, R_2\}$. Due to three *gaps* on ∂R_1, marked as dotted lines within long rectangles, $P_1 \subset \partial R_1$ has three *segments* (or maximal connected components); $P_2 = \partial R_2$ has a single segment with no gap.

There are n indistinguishable point robots residing in $\mathcal{W}$. These robots are to be deployed to *cover* the perimeters $\mathcal{P}$ such that each robot $1 \leq j \leq n$ is assigned a continuous closed subset C_j of some $\partial R_i, 1 \leq i \leq m$. All of $\mathcal{P}$ must be *covered* by $\mathcal{C} = \{C_1, \ldots, C_n\}$, i.e., $\bigcup_{P_i \in \mathcal{P}} P_i \subset \bigcup_{C_j \in \mathcal{C}} C_j$, which implies that elements of $\mathcal{C}$ need not intersect on their interiors. Hence, it is assumed that any two elements of $\mathcal{C}$ may share at most their endpoints. Such a $\mathcal{C}$ is called a *cover* of $\mathcal{P}$.

Given a cover $\mathcal{C}$, for a $C_j \in \mathcal{C}$, $1 \leq j \leq n$, let $len(C_j)$ denote its length (more formally, measure). It is desirable to minimize the maximum $len(C_j)$, i.e., the goal is to find a cover $\mathcal{C}$ such that the value $\max_{C_j \in \mathcal{C}} len(C_j)$ is minimized. This corresponds to minimizing the maximum workload for each robot or agent. The formal definition of the Optimal Perimeter Guarding (OPG) problem is provided as follows.

Problem 1 (Optimal Perimeter Guarding (OPG))**.** *Given the perimeter $\mathcal{P} = \{P_1, \ldots, P_m\}$ of a set of 2D regions $\mathcal{R} = \{R_1, \ldots, R_m\}$, find a set of n polygonal chains $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$ such that $\mathcal{C}^*$ covers $\mathcal{P}$, i.e.,*

$$\bigcup_{P_i \in \mathcal{P}} P_i \subset \bigcup_{C_j^* \in \mathcal{C}^*} C_j^*, \tag{1}$$

with the maximum of $len(C_j^*), 1 \leq j \leq n$ minimized, i.e., among all covers $\mathcal{C}$ satisfying (1),

$$\mathcal{C}^* = \underset{\mathcal{C}}{\operatorname{argmin}} \max_{C_j \in \mathcal{C}} len(C_j). \tag{2}$$

Here, we introduce the technical assumption that the ratio between the length of $\partial\mathcal{R}$ and the length of $\partial\mathcal{P}$ is polynomial in the input parameters. That is, the length of $\partial\mathcal{R}$ is not much larger than the length of $\partial\mathcal{P}$. The assumption makes intuitive sense as any gap should not be much larger than the perimeter in practice. We note that the assumption is not strictly necessary but helps simplify the correctness proof of some algorithms.

Henceforth, in general, $\mathcal{C}^*$ is used when an optimal cover is meant whereas $\mathcal{C}$ is used when a cover is meant. We further define the optimal single robot coverage length as

$$\ell^* = \min_{\mathcal{C}} \max_{C_j \in \mathcal{C}} len(C_j). \tag{3}$$

Fig. 1 shows an example of an optimal cover by 8 robots of a perimeter with three components. Note that one of the three gaps (the one on the top area as part of the hexagon) is fully covered by a robot, which leads to a smaller ℓ^* as compared to other feasible solutions. This interesting phenomenon, which is actually a main source of the difficulty in solving OPG, is explored more formally in Section III (Proposition 3).

Given the OPG formulation, additional details on $\partial\mathcal{R}$ must be specified to allow the precise characterization of the computational complexity (of any algorithm developed for OPG). For this purpose, it is assumed that each $\partial R_i \in \partial\mathcal{R}$, $1 \le i \le m$, is a simple (i.e., non-intersecting and without holes) polygon with an input complexity $O(M_i)$, i.e., ∂R_i has about M_i vertices or edges. If an OPG has a single region R, then let ∂R have an input complexity of M. Note that the algorithms developed in this work apply to curved boundaries equally well, provided that the curves have similar input complexity and are given in a format that allow the computation of their lengths with the same complexity. Alternatively, curved boundaries may be approximated to arbitrary precision with polygons.

For deploying a robot to guard a C_j, one natural choice is to send the robot to a target location $t_j \in C_j$ such that t_j is the centroid of C_j. Since C_j is one dimensional, t_j is the center (or midpoint) of C_j. After solving an OPG, there is the remaining problem of assigning the n robots to the centers of $\mathcal{C}^* = \{C_j^*\}$ and actually moving the robots to these assigned locations. As a secondary objective, it may also be desirable to provide guarantees on the execution time required for deploying the robots to reach target guarding locations. We note that, the task assignment (after determining target locations) and motion planning component for handling robot deployment, essential for applications but not a key part of this work's contribution, is briefly addressed in Section V.

With some $\mathcal{C}^*$ satisfying (1) and (2), we may further require that $len(C_j^*)$ is minimized for all $C_j^* \in \mathcal{C}^*$. This means that a gap $G \subset ((\bigcup \partial R_i)\backslash(\bigcup P_i))$ will never be partially covered by some $C_j^* \in \mathcal{C}^*$. In the example from Fig. 2, G may be one of the gaps on ∂R_1; clearly, it is not beneficial to have some C_j^* partially cover (i.e., intersect the interior of) one of these. This rather useful condition (note that this is not an assumption but a solution property) yields the following lemma.

Lemma 1. *For a set of perimeters $\mathcal{P} = \{P_1, \ldots, P_m\}$ where $P_i \subset \partial R_i$ for $1 \le i \le m$, there exists an optimal cover $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$ such that, for any gap (or maximal connected component) $G \subset ((\bigcup \partial R_i)\backslash(\bigcup P_i))$ and any $C_j^* \in \mathcal{C}^*$, $C_j^* \cap G = G$ or $C_j^* \cap G = \varnothing$.*

Remark. Our definition of coverage is but one of the possible models of coverage. The definition restricts a robot deployed to $C_j, 1 \le j \le n$, to essentially *live* on C_j. The definition models scenarios where a guarding robot must travel along C_j, which is one-dimensional. Nevertheless, the algorithms developed for OPG have broader applications. For example, subroutines in our algorithms readily solve the problem of finding the minimum number of guards needed if each guard has a predetermined maximum coverage. △

III. Structural Analysis

In designing efficient algorithms, the solution structure of OPG induced by the problem formulation is first explored, starting from the case where there is a single region.

A. Guarding a Single Region

Perimeter with a single connected component. For guarding a single region $\mathcal{R} = \{R\}$, i.e., there is a single boundary ∂R to be guarded, all n robots can be directly allocated to ∂R. If the single perimeter $P \subset \partial R$ further has a single connected component that is either homeomorphic to $\mathbb{S}^1$ or $[0, 1]$, then each robot j can be assigned a piece $C_j \subset P$ such that $\bigcup_{C_j \in \mathcal{C}} C_j = P$ and $len(C_j) = len(P)/n$. Clearly, such a cover $\mathcal{C}$ is also an optimal cover.

Perimeter with multiple maximal connected components. When there are multiple maximal connected components (or segments) in a single perimeter P, things become more complex. To facilitate the discussion, assume here P has q segments $S_1, \ldots, S_q$ arranged in the clockwise direction (i.e., $P = S_1 \cup \ldots \cup S_q$), which leaves q gaps $G_1, \ldots, G_q$ with G_k immediately following S_k. Fig. 3 shows a perimeter with five segments and five gaps.

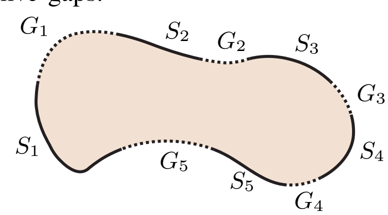

Fig. 3: A perimeter with five segments $S_1, \ldots, S_5$ and five *gaps* $G_1, \ldots, G_5$.

Suppose an optimal set of assignments for the n robots guarding P and satisfying (1) and (2) is $\mathcal{C}^* = \{C_j^*\}$. Let G_{max} be a largest gap, i.e., $len(G_{max}) = \max_{1 \le k \le q} len(G_k)$. Via small perturbations to the lengths of G_k, we may also assume that G_{max} is unique. On one hand, it must hold that $len(C_j^*) \le (len(\partial R) - len(G_{max}))/n$, as a solution where n robots evenly cover all of ∂R with the gap G_{max} excluded, satisfies the condition. On the other hand, $len(C_j^*) \ge (\sum_{1 \le k \le q} len(S_k))/n$ always holds because the coverage condition requires $\sum_j C_j^* \ge \sum_{1 \le k \le q} len(S_k)$. These yield a pair of basic upper and lower bounds for the optimal single robot coverage length ℓ^*, summarized as follows.

Proposition 2. *Define*

$$\ell_{min} = \frac{\sum_{1 \le k \le q} len(S_k)}{n} \text{ and } \ell_{max} = \frac{len(\partial R) - len(G_{max})}{n},$$

it holds that

$$\ell_{min} \le \ell^* \le \ell_{max}. \tag{4}$$

Though some gap, if there at least one, must be skipped by the optimal solution, it is not always the case that a largest gap G_{max}, even if unique, will be skipped by $\bigcup_{C_j \in \mathcal{C}^*} C_j^*$. That is, an optimal cover C^* may enclose the largest gap.

Proposition 3. *Given a region R and perimeter $P \subset \partial R$, let G_{max} be the unique longest connected component of $\partial R \backslash P$. Let $\mathcal{C}^*$ be an optimal cover of P. Then, there exist* OPG *instances in which $G_{max} \subset C_j^*$ for some $C_j^* \in \mathcal{C}^*$.*

Proof: The claim may be proved via contradiction with the example illustrated in Fig. 4 which readily generalizes. In the figure, there are four gaps $G_1, \ldots, G_4$, in which three gaps (G_1, G_2, and G_4) have the same length (i.e., $len(G_1) = len(G_2) = len(G_4)$) and are evenly spaced (i.e., $len(S_1) = len(S_2) = len(S_3 \cup G_3 \cup S_4)$). Here, $G_{max} = G_3$, which is 1.5 times the length of other gaps, i.e., $len(G_3) = \frac{3}{2} len(G_1)$.

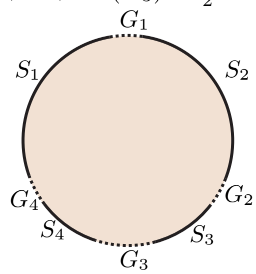

Fig. 4: A case where the perimeter has four segments or maximal connected components. Three of the gaps, G_1, G_2, and G_4 are of the same length and are evenly spaced, G_3 is 0.5 times longer.

For $n = 3$ robots, the optimal cover $\mathcal{C}^*$ must allocate each robot to guard each of S_1, S_2, and $(S_3 \cup G_3 \cup S_4)$. Without loss of generality, let $C_1^* = S_1$, $C_2^* = S_2$, and $C_3^* = (S_3 \cup G_3 \cup S_4)$. This means that G_3 is covered by C_3^* and not skipped by $\mathcal{C}^*$. In this case, $len(C_1^*) = len(C_2^*) = len(C_3^*) = len(S_1)$.

To see that this must be the case, suppose on the contrary that G_3 is skipped and let $\mathcal{C} = \{C_1, C_2, C_3\}$ be an alternative cover. By Lemma 1, an optimal cover must skip G_3 entirely. In this case, some C_j, say C_1, must have its left endpoint[1] coincide with the right endpoint of of G_3 (the point where G_3 meets S_4). Then C_1 must cover S_4 and G_4; otherwise, C_2 and C_3 must cover $S_1 \cup S_2 \cup S_3$, which makes $len(C_2) + len(C_3) \geq len(S_1 \cup S_2 \cup S_3) > 2len(S_1)$ and $\mathcal{C}$ a worse cover than $\mathcal{C}^*$. By symmetry, similarly, some C_j, say C_3, must have its right endpoint coincide with the left endpoint of G_3 and cover S_3 and G_2. However, this means that both G_2 and G_4 are covered by $\mathcal{C}$. Even if G_1 is skipped, this makes $len(C_1 \cup C_2 \cup C_3) = len(S_4 \cup G_4 \cup S_1 \cup S_2 \cup G_2 \cup S_3) > len(S_1 \cup S_2 \cup S_3 \cup G_3 \cup S_4) = 3len(S_1)$, again making $\mathcal{C}$ sub-optimal. By the pigeonhole principle, at least one of the C_1, C_2, or C_3 must be longer than $len(S_1)$. Therefore, skipping $G_{max} = G_3$ in this case leads to a sub-optimal cover. The optimal cover with $n = 3$ is to have $C^* = \{S_1, S_2, (S_3 \cup G_3 \cup S_4)\}$. ■

Proposition 3 implies that in allocating robots to guard a perimeter $P \subset \partial R$, an algorithm cannot simply start by excluding the longest component from $\partial R \backslash P$ and then the next largest, and so on. This makes solving OPG more challenging. Referring back to Fig. 1, if the top gap is skipped by the cover, then the three robots on the right side of the perimeter (two orange and one green) need to cover the part of the perimeter between the two hexagons. This will cause ℓ^* to increase.

On the other hand, for an optimal cover $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$ of P, some $C_j^* \in \mathcal{C}^*$ must have at least one of its endpoint aligned with an endpoint of a component S_k of P (assuming that $P \subsetneq \partial R$).

Proposition 4. *For an optimal cover $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$ of a perimeter $P = S_1 \cup \ldots \cup S_q \subset \partial R = S_1 \cup G_1 \cup \ldots \cup S_q \cup G_q$, for some $S_i \subset P$ and $C_j^* \in C^*$, their right (or left) endpoints must coincide.*

Proof: By Lemma 1, for any $G_k \subset \partial R \backslash P$, and $C_j^* \in \mathcal{C}^*$, $G_k \cap C_j^* = G_k$ or $G_k \cap C_j^* = \varnothing$. Since at least one G_k, $1 \leq k \leq q$, must be skipped by $C_1^* \cup \ldots C_n^*$, some C_j^*, $1 \leq j \leq n$ must have its right endpoint aligned with the right endpoint of S_k, which is on the left of G_k. Following the same argument, some $C_{j'}^*$ and $S_{k'}$ must have the same left endpoints. ■

Proposition 4 suggests that we may attempt to cover a perimeter P starting from an endpoint of S_1, S_2, and so on. Indeed, as we will show in Section IV, an efficient algorithm can be designed exploiting this important fact.

B. Guarding Multiple Regions

In a multiple region setup, there is one additional level of complexity: the number of robots that will be assigned to an individual region is no longer fixed. This introduces another set of variables $n_1, \ldots, n_m$ with $n_1 + \ldots + n_m = n$, and n_i, $1 \leq i \leq m$ being the number of robots allocated to guard ∂R_i. For a fixed n_i, the results derived for a single region, i.e., Propositions 2–4 continue to hold.

IV. Efficient Algorithms for Perimeter Guarding

In presenting algorithms for OPG, we begin with the case where each perimeter $P_i \in \mathcal{P}$ has a single connected component (i.e., P_i is homeomorphic to $\mathbb{S}^1$ or $[0, 1]$). Then, we work on the general single region case where the only perimeter is composed of $q > 1$ connected components, before moving to the most general multiple regions case.

A. Perimeters Containing Single Components

When there is a single perimeter P, the solution is straightforward with $\ell^* = len(P)/n$. With ℓ^* determined, $\mathcal{C}^*$ is also readily computed.

In the case where there are $m > 1$ regions, let the optimal distribution of the n robots among the m regions be given by $n_1^*, \ldots, n_m^*$. For a given region R_i, the n_i^* robots must each guard a length $\ell_i = len(P_i)/n_i^*$. At this point, we observe that for at least one region, say R_i, the corresponding ℓ_i must be maximal, i.e., $\ell_i = \ell^*$. The observation directly leads to a naive strategy for finding ℓ^*: for each R_i, one may simply try all possible $1 \leq n_i \leq n$ and find the maximum $len(P_i)/n_i$ that is feasible, i.e., $n - n_i$ robots can cover all other $R_{i'}$, $i' \neq i$, with each robot covering no more than $len(P_i)/n_i$. Denoting this candidate cover length $len(P_i)/n_i$ as ℓ_i^c and the corresponding n_i as n_i^c, the smallest ℓ_i^c overall $1 \leq i \leq m$ is then ℓ^*.

[1]In this paper, for a non-circular segment or gap, its left endpoint is defined as the *limit point* along the counterclockwise direction along the perimeter and its right endpoint is defined as the limit point in the clockwise direction along the perimeter. So, in Fig. 4, for S_1, its left endpoint touches G_4 and its right endpoint touches G_1.

The basic strategy mentioned above works and runs in polynomial time. It is possible to carry out the computation much more efficiently if the longest P_i is examined first. Without loss of generality, assume that P_1 is the longest perimeter, i.e., $len(P_1) \geq len(P_i)$ for all $1 \leq i \leq m$. Recall that n_1^c is the number of robots allocated to P_1 that yields ℓ_1^c, it must hold that

$$\frac{len(P_1)}{n_1^c + 1} < \ell^* \leq \frac{len(P_1)}{n_1^c} = \ell_1^c. \tag{5}$$

For an arbitrary P_i, simple manipulating of (5) yields

$$\frac{len(P_i)}{(n_1^c + 1)\frac{len(P_i)}{len(P_1)}} < \ell^* \leq \frac{len(P_i)}{n_1^c \frac{len(P_i)}{len(P_1)}}. \tag{6}$$

This means that we only need to consider $n_i^c \in [\lceil n_1^c \frac{len(P_i)}{len(P_1)} \rceil, \lfloor (n_1^c + 1)\frac{len(P_i)}{len(P_1)} \rfloor]$. Moreover, since P_1 is the longest perimeter, $\frac{len(P_i)}{len(P_1)} \leq 1$. Therefore, the difference between the two denominators of (6) is no more than 1, i.e.,

$$(n_1^c + 1)\frac{len(P_i)}{len(P_1)} - n_1^c \frac{len(P_i)}{len(P_1)} \leq 1.$$

When $len(P_i) \neq len(P_1)$, $(n_1^c + 1)\frac{len(P_i)}{len(P_1)} - n_1^c \frac{len(P_i)}{len(P_1)} < 1$ and there are two possibilities. One of these is $\lceil n_1^c \frac{len(P_i)}{len(P_1)} \rceil = \lfloor (n_1^c + 1)\frac{len(P_i)}{len(P_1)} \rfloor$, which leaves a single possible candidate for n_i^c. The other possibility is $\lceil n_1^c \frac{len(P_i)}{len(P_1)} \rceil = \lfloor (n_1^c + 1)\frac{len(P_i)}{len(P_1)} \rfloor + 1$, in which case there is actually no valid candidate for n_i^c. That is, after computing n_1^c and ℓ_1^c, if $len(P_i) = len(P_1)$ then no computation is needed for P_i. If $len(P_i) < len(P_1)$ then we only need to check at most one candidate for n_i^c.

Additional heuristics can be applied to reduce the required computation. First, in finding n_1^c, we may use bisection (binary search) over $[1, m]$ since if a given n_1 is infeasible, any $n_1' > n_1$ cannot be feasible either because $len(P_1)/n_1 < len(P_1)/n_1'$. Second, let $\ell = (\sum_{1 \leq i \leq m} len(P_i))/n$, it holds that $\ell_i^c \geq \ell^* \geq \ell$. This means that for each $1 \leq i \leq m$, it is not necessary to try any $n_i > \lfloor \frac{len(P_i)}{\ell} \rfloor$. Third, if a candidate ℓ_i^c is at any time larger than the current candidate for ℓ^*, that i does not need to be checked further. We only use the first and the third in our implementation since the second does not help much once the bisection step is applied. The pseudo code is outlined in Algorithm 1. Note that we assume the problem instance is feasible ($n \geq m$), which is easy to check.

It is straightforward to verify that Algorithm 1 runs in time $O(m \log n + m^2)$. The $O(m \log n)$ comes from the **while** loop, which calls the function ISFEASIBLE(ℓ_i^c, n_i^c, i) $\log n$ times. The function checks whether the current ℓ_i^c is feasible for perimeters other than P_i (note that it is assumed that ISFEASIBLE($\cdot$) has access to the input to Algorithm 1 as well). This is done by computing for $i' \neq i$, $n_{i'} = \lceil len(P_{i'})/\ell_i^c \rceil$ and checking whether $\sum_{i' \neq i} n_{i'} \leq n - n_i^c$. The $O(m^2)$ term comes from the **for** loop. The running time of Algorithm 1 may be further reduced by noting that the **for** loop examines $(m-1)$ candidate ℓ_i^c. These ℓ_i^c can be first computed and sorted, on which bisection can be applied. This drops the main running time to $O(m(\log n + \log m))$. This second bisection is not reflected in Algorithm 1 to keep the logic and notation more straightforward. If we also consider input complexity, an additional $O(\sum_{1 \leq i \leq m} M_i)$ is needed to compute $len(P_i)$ from the raw polygonal input and an additional $O(n)$ time is needed for generating the actual locations for the n robots. The total complexity is then $O(m(\log n + \log m) + \sum_{1 \leq i \leq m} M_i + n)$.

Algorithm 1: MULTIREGIONSINGLECOMP

```
Input  : P_1, ..., P_m: each P_i a polygon or p-chain; assume
         that P_1 is a longest perimeter
         n: the number of robots
Output: ℓ*, i*: the optimal coverage and the i realizing it
ℓ* ← ∞; i* ← 1;
%Compute n_1^c and initial ℓ*.
n_1^min ← 1; n_1^max ← n; n_1^c ← 1;
while n_1^min ≠ n_1^max do
    n_1 ← ⌈(n_1^min + n_1^max)/2⌉; ℓ_1 ← len(P_1)/n_1;
    if IsFeasible(ℓ_1, n_1, 1) then
        ℓ* ← ℓ_1; n_1^c ← n_1; n_1^min ← n_1;
    else
        n_1^max ← n_1 − 1;
    end
end
%Optimize ℓ* over all 2 ≤ i ≤ m.
for i ∈ {2, ..., m} do
    n_i^- = ⌈n_1^c len(P_i)/len(P_1)⌉; n_i^+ = ⌊(n_1^c+1) len(P_i)/len(P_1)⌋; ℓ_i ← len(P_i)/n_i^+;
    if n_i^- == n_i^+ and IsFeasible(ℓ_i, n_i^+, i) and ℓ_i < ℓ*
      then
        ℓ* ← ℓ_i; i* ← i;
    end
end
return ℓ*, i*
```

B. Single Perimeter Containing Multiple Components

Additional structural analysis. In computing ℓ^* for a single perimeter P with multiple connected components, assume that P is composed of q maximal connected components $S_1, \ldots, S_q$ (e.g., Fig. 3), leaving $G_1, \ldots, G_q$ as the gaps on ∂R. Given an optimal cover $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$, by Proposition 4, we may assume that the left endpoint of some C_j^*, $1 \leq j \leq n$ coincides with the left endpoint of some S_k, $1 \leq k \leq q$. We then look at the right endpoint of C_j^*. If it does not coincide with the right endpoint of some $S_{k'}$ (k and k' may or may not be the same), it must coincide with the left endpoint of C_{j+1}^*. Continuing like this, eventually we will hit some $C_{j'}^*$ where the right endpoint of $C_{j'}^*$ coincides with the right endpoint of some $S_{k'}$. Within a partitioned subset $C_j^*, \ldots, C_{j'}^*$, the maximal coverage of each robot is minimized when $len(C_j^*) = \ldots = len(C_{j'}^*)$. Because $\ell^* = len(C_j^*)$ for some $1 \leq j \leq n$, at least one of the subsets must have all robots cover exactly a length of ℓ^*. These two key structural observations are summarized as follows.

Theorem 5. *Let $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$ be a solution to an* OPG *instance with a single perimeter $P = S_1 \cup \ldots \cup S_q$ and gaps $G_1, \ldots, G_q$. Then, $\mathcal{C}^*$ may be partitioned into disjoint subsets with the following properties*

1) *the union of the individual elements from any subset forms a continuous p-chain,*

2) *the left endpoint of such a union coincides with the left endpoint of some S_k, $1 \leq k \leq q$,*
3) *the right endpoint of such a union coincides with the right endpoint of some $S_{k'}$, $1 \leq k' \leq q$, and*
4) *the respective unions of elements from any two subsets are disjoint, i.e., they are separated by at least one gap.*

Moreover, for at least one such subset, $\{C_j^*, \ldots, C_{j'}^*\}$, it holds that $\ell^* = len(C_j^*) = \ldots = len(C_{j'}^*)$.

In the example from Fig. 1, $\mathcal{C}^*$ is partitioned into two subsets satisfying the conditions stated in Theorem 5.

A baseline algorithm. The theorem provides a way for computing ℓ^*. For fixed $1 \leq k, k' \leq q$, denote the part of ∂R between S_k and $S_{k'}$ following a clockwise direction (with S_k and $S_{k'}$ included) as $S_{k-k'}$. Theorem 5 says that for some k, k', $len(S_{k-k'}) = n^*_{k-k'}\ell^*$ for some integer $n^*_{k-k'} \in [1, n]$. We may find k, k', and $n^*_{k-k'}$, ℓ^* by exhaustively going through all possible k, k', and $n^c_{k-k'}$ (as a candidate of $n^*_{k-k'}$). For each combination of k, k' and $n^c_{k-k'}$, we can compute a

$$\ell^c_{k-k'} = \frac{len(S_{k-k'})}{n^c_{k-k'}} \tag{7}$$

and check $\ell^c_{k-k'}$'s feasibility. The largest feasible $\ell^c_{k-k'}$ is ℓ^*.

Partial feasibility check: For checking feasibility of a particular $\ell^c_{k-k'}$, i.e., whether $\ell^c_{k-k'}$ is long enough for the rest of the robots to cover the rest of the perimeter, we simply *tile* $(n - n^c_{k-k'})$ copies $\ell^c_{k-k'}$ over the rest of the perimeter, starting from $S_{(k' \mod q)+1}$. As an example, see Fig. 5 where $n = 6$ robots are to cover the perimeter (in red, with five components $S_1, \ldots, S_5$). Suppose that the algorithm is currently working with S_{1-2} (i.e., $k = 1$ and $k' = 2$). If $n^c_{1-2} = 2$, then $\ell^c_{1-2} = len(S_{1-2})/2$. Each of the five green line segments $C_1, \ldots, C_5$ in the figure has this length. As visualized in the figure, it is possible to cover $P \backslash S_{1-2}$ with three more robots, which is no more than $n - n^c_{1-2} = 4$. Therefore, this ℓ^c_{1-2} is feasible; note that it is not necessary to exhaust all $n = 6$ robots. In the figure, C_3 covers the entire S_3 and G_3, as well as part of S_4. The rest of S_4 is covered by C_4. As C_4 is tiled, it ends in the middle of G_4, so C_5 starts at the beginning of S_5. On the other hand, if $n^c_{1-2} = 3$, the resulting ℓ^c_{1-2} (each of the orange line segments has this length) is infeasible as S_5 is now left uncovered.

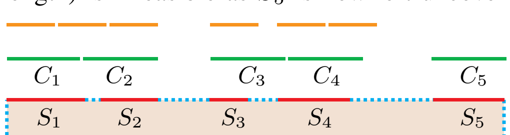

Fig. 5: An illustration of the feasibility check of ℓ^c_{1-2}. The single rectangular region and the perimeter (five red segments S_1–S_5) are shown at the bottom. The orange and green line segments show two potential covers.

The tiling-based feasibility check takes $O(q)$ time as there are at most q segments to tile; it takes constant time to tile each using a given length. Let us denote this feasibility check IsTilingFeasiblePartial(k, k', $n^c_{k-k'}$), we have obtained an algorithm that runs in $O(nq^3)$ times since it needs to go through all $1 \leq k \leq q$, $1 \leq k' \leq q$, and $1 \leq n^c_{k-k'} \leq n$. For each combination of k, k', and $n^c_{k-k'}$, it makes a call to IsTilingFeasiblePartial($\cdot$). While a $O(nq^3)$ running time is not bad, we can do significantly better.

A much faster algorithm. In the baseline algorithm, for each $k - k'$ combination, up to n candidate $n^c_{k-k'}$ may be attempted. To gain speedups, the first phase of the improved algorithm reduces the range of ℓ^* to limit the choice of $n^c_{k-k'}$. For the faster algorithm, a new feasibility checking routine is needed.

Full feasibility check: We introduce a feasibility check given only a length ℓ. That is, a check is done to see whether n robots are sufficient for covering P without any covering more than length ℓ. This feasibility check is performed in a way similar to IsTilingFeasiblePartial($\cdot$) but now k and k' are not specified. We instead try all S_k, $1 \leq k \leq q$ as the possible starting segment for the tiling. Let us denote this procedure IsTilingFeasibleFull(ℓ), which runs in $O(q^2)$.

Using bisection to limit the search range for ℓ^*: Starting from the initial bounds for ℓ^* given in Proposition 2 and with IsTilingFeasibleFull(ℓ), we can narrow the bound to be arbitrarily small, using bisection, since ℓ^* is the minimum feasible ℓ. To do this, we start with ℓ as the middle point of initial lower bound ℓ_{min} and upper bound ℓ_{max}, and run IsTilingFeasibleFull(ℓ). If ℓ is feasible, the upper bound is lowered to ℓ. Otherwise, the lower bound is raised to ℓ. In doing this, our goal in the first phase of the faster algorithm is to reduce the range for ℓ^* so that there is at most a single choice for $n^c_{k-k'}$, regardless of the values of k and k'. The stopping criteria for the bisection is given as follows, the proof of which can be found in [37].

Proposition 6. *Assume that the bisection search stops with lower and upper bound being ℓ^f_{min} and ℓ^f_{max}. If*

$$\ell^f_{max} - \ell^f_{min} < \frac{\left[\sum_{1 \leq k \leq q} len(S_k)\right]^2}{n^2 len(\partial R)}, \tag{8}$$

then there is at most a single choice for $n^c_{k-k'}$ for all k, k'.

Finding ℓ^*: Equation (8) gives the stopping criteria used for refining the bounds for ℓ^*. After completing the first phase, the algorithm moves to the second phase of actually pinning down ℓ^*. In this phase, instead of checking $\ell^c_{k-k'}$ one by one, we collect $\ell^c_{k-k'}$ for all possible combinations of k, k'. Because the first phase already ensures for each k, k' combination there is at most one pair of $n^c_{k-k'}$ and $\ell^c_{k-k'}$, there are at most q^2 total candidates. After all candidates are collected, they are sorted and another bisection is performed over these sorted candidates. Feasibility check is done using IsTilingFeasiblePartial($\cdot$). The complete algorithm is given in Algorithm 2. Note that ℓ^{min} and ℓ^{max}, which change as the algorithm runs, are not the same as the fixed ℓ_{min} and ℓ_{max} from Proposition 2.

In terms of running time, the first **while** loop starts with $\ell^{max} - \ell^{min} = \frac{len(\partial R) - len(G_{max})}{n} - \frac{\sum_{1 \leq k \leq q} len(S_k)}{n} \leq \frac{len(\partial R)}{n}$ and stops when $\ell^{max} - \ell^{min} \leq \frac{[\sum_{1 \leq k \leq q} len(S_k)]^2}{n^2 len(\partial R)}$. Therefore, the bisection is executed $\log \frac{n[len(\partial R)]^2}{[\sum_{1 \leq k \leq q} len(S_k)]^2}$ times, which by the assumption that $len(\partial R)$ is a polynomial factor over $\sum_{1 \leq k \leq q} len(S_k)$, is $O(\log(n+q))$. Since each feasibility check takes $O(q^2)$ time, the first **while** loop takes $O(q^2 \log(n+q))$ time. The **for** loops work with a total of $O(q^2)$ candidates and must sort them, taking time $O(q^2 \log q^2) = O(q^2 \log q)$. Then, the second **while** loop bisects $O(q^2)$ candidates and calls IsTilingFeasiblePartial($\cdot$) for each check, taking time $O(q \log q^2) = O(q \log q)$. The total running time of Algorithm 2 is then $O(q^2 \log(n+q) + M + n)$.[2]

Algorithm 2: SingleRegionMultiComp

Input : $\partial R = S_1 \cup G_1 \cup \ldots \cup S_q \cup G_q$: a single boundary with the perimeter $P = S_1 \cup \ldots \cup S_q$. n: the number of robots

Output: ℓ^*, k^*, k'^*: the optimal coverage and the pair of k and k' that realize the optimal coverage

%Phase one: narrow the range of ℓ^*.

$\ell^{min} \leftarrow \frac{\sum_{1\le k\le q} len(S_k)}{n}$, $\ell^{max} \leftarrow \frac{len(\partial R) - len(G_{max})}{n}$;

while $\ell^{max} - \ell^{min} > \frac{[\sum_{1\le k\le q} len(S_k)]^2}{n^2 len(\partial R)}$ **do**

  $\ell \leftarrow \frac{\ell^{max}+\ell^{min}}{2}$;

  (IsTilingFeasibleFull(ℓ)? $\ell^{max} \leftarrow \ell$: $\ell^{min} \leftarrow \ell$);

end

%Phase two: pin down ℓ^*.

$sm \leftarrow []$; %sm is a sorted map.

for $k, k' \in \{1, \ldots, q\}$ **do**

  $n^{max}_{k-k'} \leftarrow \lfloor \frac{len(S_{k-k'})}{\ell^{min}} \rfloor$; $n^{min}_{k-k'} \leftarrow \lceil \frac{len(S_{k-k'})}{\ell^{max}} \rceil$;

  for $n^c_{k-k'} \in \{n^{min}_{k-k'}, \ldots, n^{max}_{k-k'}\}$ **do**

    sm.put($\frac{len(S_{k-k'})}{n^c_{k-k'}}$, $(n^c_{k-k'}, \frac{len(S_{k-k'})}{n^c_{k-k'}}, k, k')$);

  end

end

$\ell^* \leftarrow \infty$; $k^* \leftarrow 0$; $k'^* \leftarrow 0$;

while sm.size() > 1 **do**

  %Extract the element from sm in the middle.

  $(n^c, \ell^c, k, k') \leftarrow sm$.middleValue();

  if IsTilingFeasiblePartial(k, k', n^c) **then**

    $\ell^* \leftarrow \ell^c$; $k^* \leftarrow k$; $k'^* \leftarrow k'$;

    $sm \leftarrow sm$.range(sm.minKey(), ℓ^c);

  else

    $sm \leftarrow sm$.removeRange(sm.minKey(), ℓ^c);

  end

end

return ℓ^*, k^*, k'^*

C. Multiple Perimeters Containing Multiple Components

The algorithm for the multiple perimeter case is a direct generalization Algorithm 2. To facilitate the description, let the perimeter P_i, $1 \le i \le m$, contain q_i maximal connected components, i.e., $P_i = S_{i,1} \cup \ldots \cup S_{i,q_i}$ and the boundary $\partial R_i = S_{i,1} \cup G_{i,1} \cup \ldots \cup S_{i,q_i} \cup G_{i,q_i}$. We extend the definition of $S_{k-k'}$ for a single perimeter to $S_{i,k-k'}$ for multiple perimeters. By a straightforward generalization of Theorem 5 to multiple perimeters, for an OPG instance, the length of some $S_{i,k-k'}$ must be an integer multiple of ℓ^*. Similar to the single perimeter case, we can try all $S_{i,k-k'}$ and for each try all possible $1 \le n^c_{i,k-k'} \le n$. This gives us $\ell^c_{i,k-k'} = \frac{len(S_{i,k-k'})}{n^c_{i,k-k'}}$ as candidates for ℓ^*; there are $n(\sum_{1\le i\le m} q_i^2)$ such candidates. For checking the feasibility of $\ell^c_{i,k-k'}$, we may use IsTilingFeasiblePartial($\cdot$) for the rest of P_i (taking $O(q_i)$ time) and IsTilingFeasibleFull($\cdot$) for all $1 \le i' \le m$ and $i' \ne i$ (taking $O(\sum_{1\le i'\le m, i'\ne i} q_{i'}^2)$ time). This yields a baseline algorithm that runs in $O(n(\sum_{1\le i\le m} q_i^2)^2)$ time.

From here, speedups can be obtained as in the single perimeter case using the same reasoning. This yields a two-phase algorithm, which we call MultiRegionMultiComp, that runs in $O((\sum_{1\le i\le m} q_i^2)\log(n + \sum_{1\le i\le m} q_i) + \sum_{1\le i\le m} M_i + n)$.

[2]We note that the assumption that $len(\partial R)$ is a polynomial factor over $\sum_{1\le k\le q} len(S_k)$ is not necessary. However, the corresponding analysis becomes much more involved without it. Since the assumption makes practical sense and also due to space limit, the more general result is omitted from the current paper. Many additional interesting but non-essential details, including this one, will be included in an extended version.

V. Performance Evaluation and Applications

Our evaluation first verifies the algorithms' running time matches the claimed bounds. Then, two practical scenarios are illustrated to show how OPG may be adapted to applications.

A. Algorithm Performance

In the performance results presented here, a data point is the average from 10 randomly generated OPG instances. All algorithms are implemented in Python 2.7, and all experiments are executed on an Intel® Xeon® CPU at 3.0GHz.

For the case of m perimeters each containing a single segment, for each $1 \le i \le m$, we set $len(\partial R_i) = 1$ and let $len(P_i)$ be uniformly distributed in $(0, 1]$. Fig. 6 shows the result for an example with $m = 10$ and $n = 30$. For various values of m, n, the running time of MultiRegionSingleComp is summarized in Table I, which scales very well with m and n (note that the $n \le m$ case does not make much sense here).

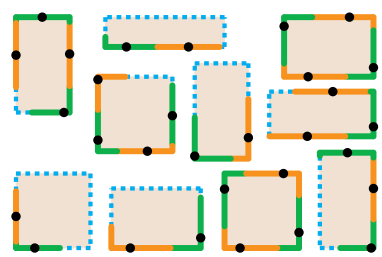

Fig. 6: An example problem instance when $m = 10$ and $n = 30$. The black dots indicate deployed robot locations; the green and orange p-chains indicate the coverage.

m \ n	10^8	10^9	10^{10}	10^{11}	10^{12}
10^6	1.152	1.442	1.508	1.652	1.617
10^7	13.963	17.281	18.796	20.354	20.627
10^8	NA	176.115	223.186	227.250	230.000

TABLE I. MultiRegionSingleComp running time (seconds)

For the case of a single perimeter with multiple components, a random polygon is generated on which $2q$ points are randomly sampled that yield q segments (that form the perimeter) and q gaps. An example instance and the optimal solution with $q = 3$ and $n = 10$ is illustrated in Fig. 7. The computation time for various q and n combinations is given in Table II.

For multiple perimeters containing multiple components, m polygons are created with $len(\partial R_i)$ randomly distributed in $[1, 10]$. For setting q_i, we fix a q and let $q_i = q(0.5 + random(0,1))$. Representative computation results of MultiRegionMultiComp are listed in Table III.

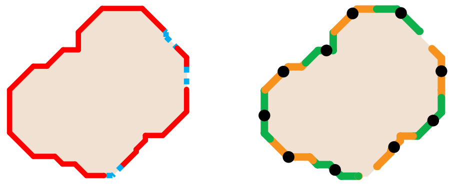

Fig. 7: An example problem instance when $q = 3$ and $n = 10$. In this case, the optimal cover actually covers one gap.

q \ n	10^1	10^2	10^3	10^4	10^5
10^2	0.013	0.015	0.016	0.016	0.017
10^3	1.363	1.595	1.622	1.634	1.641
10^4	159.404	188.497	210.492	212.473	212.780

TABLE II. SINGLEREGIONMULTICOMP computation time (seconds)

q	n	m				
		10	20	30	40	50
10^1	10^3	0.047	0.063	0.076	0.091	0.108
10^2	10^3	2.191	3.771	5.523	7.707	9.369
10^2	10^4	7.105	9.619	11.369	12.760	15.107

TABLE III. MULTIREGIONMULTICOMP computation time (seconds)

Due to limited space, only selected essential performance data is presented here. More complete performance data and associate analysis can be found [37].

B. Two Applications Scenarios

Securing a perimeter. As a first application, consider a situation where a crime has just been committed at the Edinburgh Castle (see Fig. 8). The culprit remains in the confines of the castle but is mixed within many guests at the scene. As the situation is being investigated and suppose that the brick colored buildings are secured, guards (either personnel or a number of drones) may be deployed to ensure the culprit does not escape by climbing down the castle walls. Using SINGLEREGIONMULTICOMP, a deployment plan can be quickly computed given the amount of resources at hand so that each guard only needs to secure a minimum length along the castle walls. Fig. 8 shows the optimal deployment plan for 15 guards.

Fig. 8: Optimal deployment of 15 guards around walls of the Edinburgh Castle. The brick colored structures are buildings that create gaps along the boundary.

Fire monitoring. In a second application, consider Fig. 9 where a forest fire has just been put out in multiple regions. As there is still some chance that the fire may rekindle and spread, for prevention, a team of firefighters is to be deployed to watch for the possible spreading of the fire. Here, in addition to using MULTIREGIONMULTICOMP to compute optimal locations for deploying the firefighters, we also generate minimum time trajectories for the firefighters to reach their target locations while avoiding going through the dangerous forests. This is done via solving a bottleneck assignment problem [38]. Note that the lake region creates gaps that cannot be traveled by the firefighters; this can be handled by making these gaps infinitely large. Fig. 9 shows the optimal locations for 34 firefighters. Animations of the deployment process and other test cases can be found in the accompanying video.

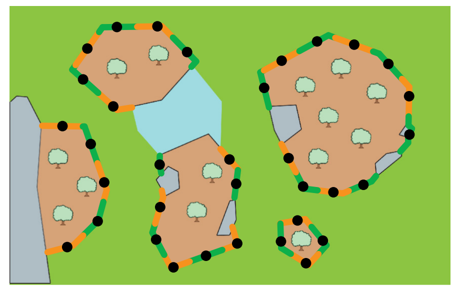

Fig. 9: Optimal deployment of 34 firefighters for forest fire rekindling prevention.

VI. CONCLUSION AND DISCUSSION

In this paper, we propose the OPG problem to model the allocation of a large number of robots to cover complex 1D topological domains with optimality guarantees. For all variants under the OPG formulation umbrella, we have developed highly efficient algorithms for solving OPG exactly. In addition to rigorous proofs backed by formal analysis, extensive computational experiments further confirm the effectiveness of these algorithms. Moreover, practical relevance of OPG is demonstrated through the integration of OPG into realistic task (assignment) and motion planning scenarios.

The study raises many additional interesting open questions; we mention a few here. First, the approach taken in this work is a *centralized* one where decision is made at the global level. It would be highly interesting to explore whether the same can be achieved with *decentralized* methods, which have many advantages. For example, it may be the case that the gaps along the boundaries are not known *a priori* and must be measured by the robots. In such cases, a centralized plan can be hard to come by. Second, as mentioned in Section II, the current OPG formulation assumes that the robots are confined to the boundaries $\partial\mathcal{R}$, which is one of many possible choices in terms of the robots' sensing and/or motion capabilities. In future study, we plan to examine additional practical robot sensing and motion models. Third, as exact optimal algorithms are emphasized here, issues including uncertainty and robustness have not been touched in the current treatment, which are important elements when it comes to the deployment of a robotic swarm to tackle real-world challenges.

Acknowledgments. This work is supported by NSF grants IIS-1617744, IIS-1734419, and IIS-1845888. Opinions or findings expressed in this paper do not necessarily reflect the views of the sponsors. The authors would like to thank the anonymous RSS reviewers for their constructive comments.

REFERENCES

[1] T. Arai, E. Pagello, and L. E. Parker, "Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[2] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[3] W. Ren and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008.

[4] F. Bullo, J. Cortes, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009, vol. 27.

[5] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed memoryless point convergence algorithm for mobile robots with limited visibility," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999.

[6] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[7] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[8] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on automatic control*, vol. 50, no. 5, pp. 655–661, 2005.

[9] P. Cheng and V. Kumar, "An almost communication-less approach to task allocation for multiple unmanned aerial vehicles," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1384–1389.

[10] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.

[11] J. Yu, S. M. LaValle, and D. Liberzon, "Rendezvous without coordinates," *IEEE Transactions on Automatic Control*, vol. 57, no. 2, pp. 421–434, 2012.

[12] S. L. Smith and F. Bullo, "Monotonic target assignment for robotic networks," *IEEE Transactions on Automatic Control*, vol. 54, no. 9, pp. 2042–2057, 2009.

[13] N. Ayanian and V. Kumar, "Decentralized feedback controllers for multiagent teams in environments with obstacles," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 878–887, 2010.

[14] L. Liu and D. A. Shell, "Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation," *Robotics: Science and Systems VII; MIT Press: Cambridge, MA, USA*, pp. 26–33, 2011.

[15] ——, "Optimal market-based multi-robot task allocation via strategic pricing." in *Robotics: Science and Systems*, vol. 9, no. 1, 2013, pp. 33–40.

[16] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[17] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[18] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot navigation in formation via sequential convex programming," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 4634–4641.

[19] K. Solovey, J. Yu, O. Zamir, and D. Halperin, "Motion planning for unlabeled discs with optimality guarantees," in *Robotics: Science and Systems*, 2015.

[20] D. E. Soltero, M. Schwager, and D. Rus, "Decentralized path planning for coverage tasks using gradient descent adaptive control," *The International Journal of Robotics Research*, vol. 33, no. 3, pp. 401–425, 2014.

[21] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[22] M. Schwager, B. J. Julian, and D. Rus, "Optimal coverage for multiple hovering robots with downward facing cameras," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3515–3522.

[23] J. O'rourke, *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987, vol. 57.

[24] T. C. Shermer, "Recent results in art galleries (geometry)," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1384–1399, 1992.

[25] V. Klee, "Is every polygonal region illuminable from some point?" *The American Mathematical Monthly*, vol. 76, no. 2, pp. 180–180, 1969.

[26] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[27] D. Lee and A. Lin, "Computational complexity of art gallery problems," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 276–282, 1986.

[28] A. Thue, *Über die dichteste Zusammenstellung von kongruenten Kreisen in einer Ebene, von Axel Thue...* J. Dybwad, 1910.

[29] T. C. Hales, "A proof of the kepler conjecture," *Annals of mathematics*, pp. 1065–1185, 2005.

[30] Z. Drezner, *Facility location: a survey of applications and methods*. Springer Verlag, 1995.

[31] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[32] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Equitable partitioning policies for robotic networks," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2356–2361.

[33] S. Martínez, "Distributed interpolation schemes for field estimation by mobile sensor networks," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 491–500, 2010.

[34] A. Pierson, L. C. Figueiredo, L. C. Pimenta, and M. Schwager, "Adapting to sensing and actuation variations in multi-robot coverage," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 337–354, 2017.

[35] R. B. Kershner, "On paving the plane," *The American Mathematical Monthly*, vol. 75, no. 8, pp. 839–844, 1968.

[36] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

[37] S. W. Feng, S. D. Han, K. Gao, and J. Yu, "Efficient algorithms for optimal perimeter guarding," *arXiv:1905.04434*, 2019.

[38] R. E. Burkard and E. Cela, "Linear assignment problems and extensions," in *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.